Feeling the Pulse of Linked Data*

Laurens Rietveld[1], Rinke Hoekstra[1,2], and Stefan Schlobach[1]

[1] Dept. of Computer Science, VU University Amsterdam, NL
{laurens.rietveld,k.s.schlobach,rinke.hoekstra}@vu.nl
[2] Leibniz Center for Law, University of Amsterdam, NL, hoekstra@uva.nl

Abstract. Exploring and analyzing the Linked Data Cloud is hard because it is often unclear where the data resides, and what vocabularies and namespaces are used to express it. Existing studies focus on the availability of data rather than its use in practice. To make matters worse, existing tools for interacting with the data are very feature poor. We use YASGUI, a feature rich web-based query editor, as a measuring device for interactions with the Linked Data Cloud. It enables us to determine what part of the Linked Data Cloud is actually used, what part is open or closed, the efficiency and complexity of queries, and how these results relate to commonly used dataset statistics.

Keywords: SPARQL, Linked Data, Semantic Web, Usage analysis

1 Introduction

As the Linked Data cloud grows both in size and complexity, it becomes increasingly interesting to study how, and what parts are being used for which purpose. For these purposes, there are currently two approaches: the study of query logs, such as provided by the USEWOD series [3], and of gathering dataset statistics [4, 11]. Both only partially fulfill their intended purpose because 1) they are restricted to a small number of datasets and 2) the information is collected at the publisher rather than the user-end of the development pipeline. What is missing for analytics over the Linked Data cloud is a dataset independent data collection point, which can act as a kind of observational lens.

Take as analogy the query logs collected by search engines, such as Google or Yahoo. These have become the primary proxies for studying information need on the World Wide Web. This has to do with the unique position those engines have as *the* central filters through which users access the otherwise distributed information. Indeed, the business model of web search giants is founded on their ability to adequately target advertisements to users, based on their search behavior. For the Web of Data, not a single such entry point currently exists. This paper uses statistics generated by YASGUI, a SPARQL client launched in early 2013, which has the potential for becoming such an observational lens for the Linked Data cloud.

* This work was supported by the Dutch national program COMMIT

YASGUI[3], first introduced in [23], is a web-based query editor for the Web of Data that uses the latest web technologies. It is packed with usability features such as auto-completion, syntax highlighting, dataset endpoint search, and sharing functionalities for SPARQL. When given permission to do so, it acts as a measuring device for Linked Data, by tracking the actions of users. This provides deep insight in how we interact with Linked Data. As YASGUI works for every SPARQL endpoint, it can collect information on more than the Linked Data cloud we were previously aware of, including endpoints inaccessible from the internet. In section 3.2 we show how the information collected through this SPARQL interface increases our knowledge of Linked Data, such as which part of the Linked Data cloud is actually used, what part is open and accessible, the complexity of man-made queries, and the most commonly used namespaces.

The matter of uptake is the critical factor as to whether or not YASGUI will eventually collect sufficient, valid, and unbiased data, and can become a proper observational lens. In Section 3.1 we argue that there are sufficient incentives for users to use it as their point of entry for the Linked Data cloud as it is the most user friendly, intuitive and interactive interface to date.

Structure of the paper This paper is structured as follows. First, in section 2 we discuss related approaches to the study of the Linked Data cloud, and we review the features present in the state of the art in SPARQL user interfaces. Section 3 outlines our methodology, and summarizes the features of YASGUI in section 3.1. Section 4 discusses how the use of YASGUI allows us to analyze the Linked Data cloud, and what we can observe from the data we gathered since its launch. We conclude in section 5.

2 Related Work

2.1 Linked Data Analysis

The most well known depiction of Linked Data is a "cloud" of 311 connected ("linked") datasets [4]. The size of circles depends on the size of the datasets, and links represent the reuse of identifiers between datasets. Not only is the latest version outdated (November 2011), it is also rather limited in that it is based on metadata that were manually registered in the Datahub CKAN catalog[4] and which have an open license. This makes the analysis quite unreliable and static: there is no check as to whether the size and number of links registered correspond to reality, and there is no indication of whether the data is actually being used.

LODStats [11] assesses the availability of the information in the Datahub. It attempts to access or download registered datasets, and extracts structure and schema characteristics. Results show that for various reasons, only a fraction of

[3] See `http://yasgui.org` (6 May 2014)

[4] See `http://datahub.io` (20 Feb. 2014)

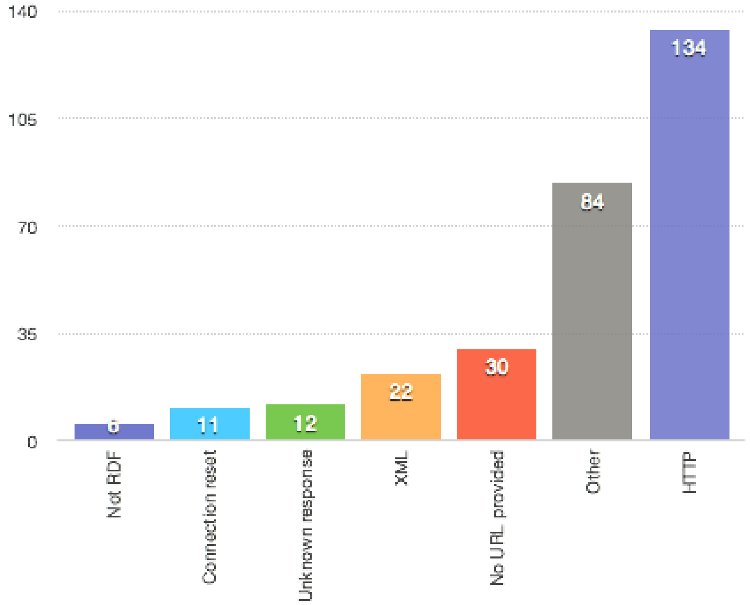


Fig. 1: Problems with the Datahub Linked Data cloud ([14])

the registered data is accessible in practice (cf. Figure 1, from [14]). [15] performed an in depth analysis of the quality of Linked Data that was crawled from the Web as part of the Billion Triple Challenge in 2011[5], focusing in particular on the adherence of the datasets to Linked Data principles such as dereferencability of URIs. These efforts show that accessibility is hampered by the reliability of services hosting the data.

SPARQLES [8] continuously tracks the uptime of SPARQL endpoints, which SPARQL features these endpoints support, and which endpoints publish dataset statistics. This tool is useful for observing the current state of accessible SPARQL endpoints, though again, the set of endpoints is limited to those published on CKAN. However, since these analyses are still based on the Datahub registry, and thus on some incentive to manually enter metadata, we cannot simply conclude that the inaccessibility of the known Linked Data cloud means that the actual size of the Linked Data cloud is grossly overstated.

Sindice [26] collects data from the Web of Data by crawling web pages for RDFa and microformat markup. It also collects data from endpoints, though this is a manual procedure, and happens on a per-request basis. As a result, Sindice provides an extensive amount of information about the Web of Data, taking a broader perspective than focusing on SPARQL endpoints alone. Because SPARQL endpoints are added on a per-request basis, the gathered data is still incomplete with eight datasets compared to the 311 datasets in the LOD cloud.

[5] See `http://km.aifb.kit.edu/projects/btc-2011/`.

To better understand the usage of Linked Data, the USEWOD [3] workshop series initiated a challenge to analyze server logs from six well known SPARQL query endpoints (datasets): DBpedia [1], Semantic Web Dog Food [19], BioPortal [20], Bio2RDF [9], Open-BioMed[6] and Linked Geo Data [2]. Clearly this only covers a small portion of the number of datasets registered in the Datahub, making it difficult to extrapolate to the full size of the Web of Data. Also, the query logs make no distinction between 'machine queries' – queries executed by applications – and manual interaction with Linked Data [18]. In previous work [24], we quantified exactly this difference, by comparing the YASGUI set of man-made queries with queries taken from server logs (containing mostly machine queries). We showed that queries from each sets differ greatly in size, the range of SPARQL features they use, and complexity.

2.2 SPARQL interfaces

Besides Linked Data that is hidden in webpages, e.g. RDFa markup using the Schema.org or GoodRelations vocabulary, by far the most expressive use of Linked Data is done through query interfaces for the SPARQL query language, that operate on web-based interfaces to triple stores. Many such SPARQL clients exist, but they lack the feature richness needed to attract sufficient numbers of users, and to study SPARQL usage across datasets. Table 1 lists fourteen currently existing SPARQL clients – that range from very basic to elaborate – and depicts what features they implement. We briefly discuss them below. The NITE-LIGHT [25], SPARQLinG[16], ViziQuer [27] and SPARQLViz [6] clients are not listed as they were no longer available at the time of writing, were unstable, or do not work in any recent operating system[7]. The YASGUI client is presented separately in Section 3.1.

SPARQL is a complex language and queries can become quite large. Syntax highlighting and checking can help significantly to improve readability of queries, but the Flint SPARQL Editor is the only client that currently supports it[8]. TopBraid Composer[9] and Flint (and indirectly, the SparQLed editor[10] based on the former) support auto-completion for suggesting classes and properties. This increases transparency, as the auto-completion may suggest information that a user was not aware of.

There are only four clients that fully support access to multiple endpoints. This is because many clients are part of the web frontend of triple stores. Examples are 4Store [13], OpenLink Virtuoso [21], OpenRDF Sesame Workbench [7] and SPARQLer[10]. More generic clients are the Sesame2 Windows Client [7],

[6] See `http://www.open-biomed.org.uk/`, (6 May 2014)

[7] Contact with the authors behind these tools was either not successful, or did not result in a reproducible tool.

[8] See `http://openuplabs.tso.co.uk/demos/sparqleditor` (21 Feb. 2014)

[9] See `http://www.topquadrant.com/` (21 Feb. 2014)

[10] See `http://www.sparql.org/` (21 Feb. 2014)

Feature	iSPARQL	4Store	OpenLink Virtuoso	SNORQL	SPARQLer	Sesame Workbench	Sesame2 Windows Client	TopBraid Composer	LODatio	Glint	Twinkle	SparqlGUI	SparQLed	Flint SPARQL Editor	YASGUI
Auto-completion	+	-	-	-	-	-	-	-	-	-	-	-	+ [a]	+ [a]	+ [b]
Syntax Highlighting	N/A	-	-	-	-	-	-	+	-	+	-	-	+	+	+
Syntax Checking	N/A	-	-	-	-	-	-	+	-	-	-	-	+	+	+
Multiple Endpoints	-	-	-	-	-	-	±	-	+	+	+	± [c]	-	± [c]	+
Platform independent	+	+	+	+	+	+	-	+	+	-	+	-	+	+	+
Full SPARQL 1.1 syntax	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Query retention	-	-	-	-	-	-	+	+	-	+	-	+	-	-	+
File upload	-	-	-	-	-	+	± [d]	+	-	-	+	+	-	-	- [e]
Results rendering	+	-	± [f]	+	± [f]	+	± [f]	+	+	± [f]	± [f]	± [f]	+	+	+
Results download	-	+	+	+	+	+	+	+	-	+	+	+	-	-	+
Visual query interface	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-

[a] Auto-completion of properties and classes available in the triple store

[b] Autocompletion of prefixes/namespaces/properties/classes

[c] Can deal with a limited number of endpoints, e.g. only CORS enabled ones.

[d] File upload requires a local triple store that implements the OpenRDF SAIL API, e.g. OpenRDF Sesame or OpenLink Virtuoso.

[e] File upload is a planned feature, using cloud triple-store services (e.g. dydra.com)

[f] The rendering does not use hyperlinks for URI resources.

Table 1: SPARQL client feature matrix

Glint[11], Twinkle[12] and SparqlGUI[13]. Other applications fall somewhere in between. The FLINT SPARQL Editor only connects to endpoints which support cross-domain JavaScript (i.e. CORS enabled). This is a problem because not all endpoints are CORS enabled, such as FactForge, CKAN, Mondeca or data.gov. Other editors support only XML or JSON as query results, such as SNORQL[14] (part of D2RQ [5]), which only support query results in SPARQL-JSON format. TopBraid composer supports querying multiple endpoints only via the the SPARQL `SERVICE` federated query functionality of SPARQL 1.1. Finally, LODatio indexes the schema from multiple datasets, but not all of them, and not all information is indexed from those that are.

The clients shipped with Virtuoso and 4Store and the Flint SPARQL Editor are Web-based and thus platform independent. Twinkle is a Java application, making it runnable on almost any operating system. Examples of single-

[11] See `https://github.com/MikeJ1971/Glint` (21 Feb. 2014)

[12] See `http://www.ldodds.com/projects/twinkle/` (21 Feb. 2014)

[13] See `http://www.dotnetrdf.org/content.asp?pageID=SparqlGUI` (21 Feb. 2014)

[14] See `https://github.com/kurtjx/SNORQL/` (21 Feb. 2014)

platform applications are Sesame2 Windows Client and SparqlGUI: they require Windows. All text-oriented clients provide complete SPARQL syntax support. This is harder to accomplish for clients with a visual query interface, such as iSPARQL.

Query retention allows for easy re-use of important or often used queries. This allows the user to close the application, and resume working on the query later. An example is the 'Query Book' functionality of the Sesame Windows Client. Exploring small RDF graphs should not necessitate the hassle of installing a local triple-store. Several applications such as Twinkle and The Sesame Windows Client support uploading of files.

The raw results to SPARQL queries are very hard to read. All applications except 4Store render the results of SELECT queries as a table. Results typically contain URIs, that invite navigation of the RDF graph. However, not all clients support it (Virtuoso, Twinkle or SparqlGUI). SNORQL allows users to navigate to the Web address of the URI, or the user can click on a link to browse the current endpoint for resources relevant to that URI. Finally, it can be useful to be able to download the results to SPARQL queries, e.g. the results of CONSTRUCT queries are often used in other applications. The only applications that do not support the downloading of results are the FLINT SPARQL editor and SparQLed.

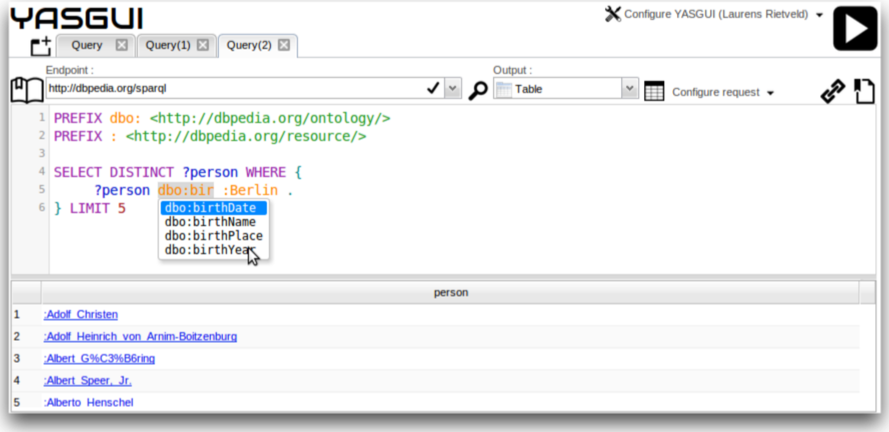
3 Methodology

The discussion of related work shows that we can only sketch a reliable picture of the Linked Data cloud that includes both the presence and use of datasets if we tap into where interaction with the Linked Data cloud occurs: on the client side. Our method follows two steps, we 1) developed a SPARQL client (YASGUI) that can attract users and allows access to all SPARQL endpoints (Section 3.1), we then 2) ask permission to log user queries, and analyze these queries along various dimensions such as namespaces, endpoints, complexity, etc. (Section 3.2). The results of this analysis are discussed in 4

3.1 The Features of YASGUI

YASGUI is a knife that cuts on both sides: it is a tool that makes it easier to interact with Linked Data, and it allows us to gather an unprecedented wealth of usage data if users opt-in. We argue that it is the most complete SPARQL client available, containing unique additional features for auto-completion and collaborative editing, which have not been available in SPARQL interfaces before. We introduced this tool in [23], but will briefly discuss its features here.

YASGUI supports syntax highlighting and checking (like FLINT) but it provides extensive auto-completion features as well: auto-completion of properties and classes are supported, and full namespace URIs of prefixes are added as you type. It supports access to any SPARQL endpoint, and provides auto-completion



Endpoint : http://dbpedia.org/sparql

Output : Table

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX : <http://dbpedia.org/resource/>

SELECT DISTINCT ?person WHERE {
     ?person dbo:bir :Berlin .
} LIMIT 5
```

	person
1	:Adolf_Christen
2	:Adolf_Heinrich_von_Arnim-Boitzenburg
3	:Albert_G%C3%B6ring
4	:Albert_Speer,_Jr.
5	:Alberto_Henschel

Fig. 2: Screenshot of the YASGUI interface

and searching for endpoints using the CKAN SPARQL endpoint[15] and SPARQLES [8][16]. The FLINT client is also endpoint independent, but only works for endpoints that allow Cross-Origin Resource Sharing (CORS).[17] YASGUI implements a proxy that allows access to all CORS disabled endpoints. For this reason, YASGUI supports the specification of an arbitrary number of request parameters that are sent along the HTTP request (e.g. the 'soft-limit' parameter of 4Store). YASGUI allows query results to be downloaded as CSV or 'as is' (for raw query results). It provides a tabular view of query results that allows users to browse the Web of Data through clicking on resource URIs.

The YASGUI application state is persistent across sessions: a returning user will see the screen as it was when she last closed the YASGUI browser page. Queries can be bookmarked, and connected to an OpenID account. This way users are able to re-use queries between user sessions, browsers, and computers. Furthermore, YASGUI can generate a permalink for each query. Opening the link in a browser opens YASGUI with the specified query, endpoint and request arguments filled in. We believe this is a welcome feature for people working together with a need to share queries. Finally, YASGUI can be used offline, as a regular desktop application, by means of the HTML5 offline manifest functionality.

3.2 Analysis

Since the public launch of YASGUI 1 year ago, it has been quite successful in attracting visitors from across the world (See Figure 3): 2947 unique visitors

[15] See `http://datahub.io` (6 May 2014)

[16] Unfortunately, the former is not frequently updated, and the latter has been taken offline.

[17] See `http://www.w3.org/TR/cors/` (21 Feb. 2014)

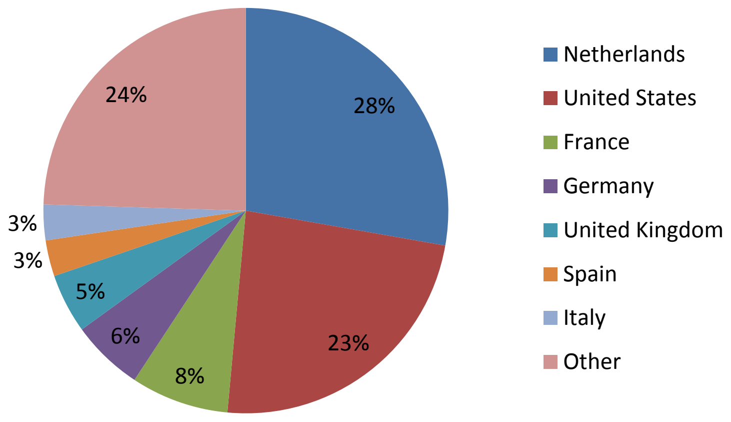


Fig. 3: Location of YASGUI users

to the website, and 45.000 queries were executed against 793 endpoints. This section elaborates on the data we can gather from these visitors, the types of analysis we run on the data, and some observations that can be made (section 4). We use Google Analytics[18] to log the actions of users that explicitly allow us to do so: every user is presented with an opt-out form in which users may choose to disable logging completely, or to disable logging of endpoints and queries only. Until now, 58% of our users allow full logging, 6% disabled logging of endpoints and queries only, where the remainder disabled logging altogether. User actions include the queries a user executes, the endpoint they use, the time it takes to get the query response[19], and more general information such as (an estimate of) the user's location and the local time.

Given these logs we can study the following:

1. How do the SPARQL endpoints registered in CKAN relate to the endpoints used in YASGUI? How big is the overlap?
2. Looking at the datasets hosted by these endpoints, what part of the dataset is actually needed to answer the queries posed against it?
3. What namespaces are most commonly used in the the queries?
4. How complex are the queries, how many are there?

Our analysis of the query complexity is a two-level approach. At a general level, we extract relatively simple information such as the use of SPARQL keywords (e.g. DESCRIBE, ASK, SELECT). At the a more finegrained level, we

[18] See `http://www.google.com/analytics/` (6 May 2014)

[19] Logging the execution time of queries is added recently. Therefore, these results are not included in this paper

Total Queries	45.323
Valid Queries	30.482
Unique Queries	18.162
Type	
`SELECT`	94.52%
`DESCRIBE`	0.74%
`ASK`	1.59%
`CONSTRUCT`	3.15%
`INSERT`	0.00%
Complexity	
≥ 1 joins	54.35%
≥ 1 V C C pattern	58.37%
≥ 1 V C V pattern	53.68%
≥ 1 C C V pattern	11.92%
≥ 1 C V V pattern	10.44%
≥ 1 V V C pattern	9.87%
≥ 1 V V V pattern	7.76%
≥ 1 C C C pattern	0.96%
≥ 1 C V C pattern	0.30%

(a) Queries

	#	Weighted by #queries
Accessible endpoints		
CKAN endpoints	84	73.45%
Not in CKAN	124	7.61%
Inaccessible endpoints		
Probably incorrect	447	1.22%
Contains private data	105	11.02%
Only contains public data	171	6.70%

(b) Endpoints

Fig. 4: LD Usage Properties

analyze the complexity of the query sets, using the methods described in [12, 22]. We look at two aspects: the triple pattern structure and the number of joins. The number of triple patterns used in queries, as well as the structure of these triple patterns is a good indication of the complexity of queries. We use the method described in [12] to determine types of joins, and the number of joins per query. Each element in a triple can be a variable (V), or a constant (C). For instance, `[] rdf:type ?object` can be classified as `V C V`. When two triple patterns have one variable in common, the query engine would need to join both.

4 Results

Endpoint Usage In the well known Linked Open Data cloud [4], edges between datasets indicate links between resources of both datasets, and the node size indicates the number of triples of the dataset. As mentioned in section 2, this dataset catalog has strong curation limitations. With our logs, we can observe the actual usage of endpoints in the LOD-cloud. The datasets in Figure 5 are retrieved from CKAN using the same code of the original LOD diagram[20]. The gray endpoints have never been accessed via YASGUI, where the blue nodes are, indicating that only a small part (20%) of the endpoints in this particular LOD-cloud representation is used. Note, however, that not all datasets in CKAN are hosted online, which means that this overlap may be larger in reality.

[20] See `https://github.com/lod-cloud/datahub2void` (6 May 2014)

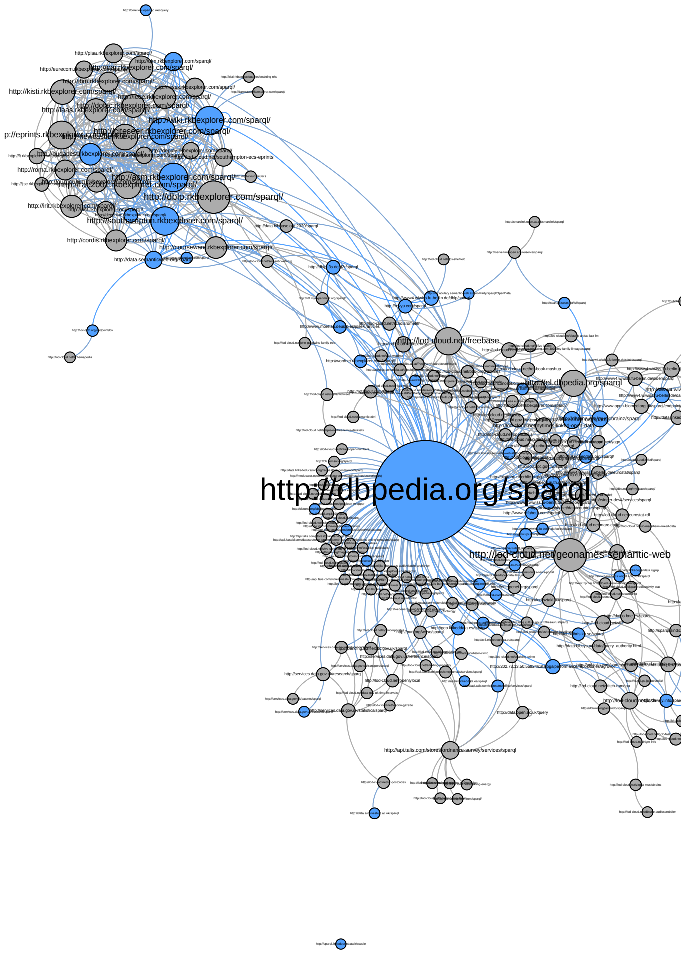

http://dbpedia.org/sparql
http://lod-cloud.net/freebase
http://el.dbpedia.org/sparql
http://lod-cloud.net/geonames-semantic-web
http://dblp.rkbexplorer.com/sparql/
http://southampton.rkbexplorer.com/sparql/
http://wiki.rkbexplorer.com/sparql/
http://citeseer.rkbexplorer.com/sparql/
http://acm.rkbexplorer.com/sparql/
http://rae2001.rkbexplorer.com/sparql/
http://kisti.rkbexplorer.com/sparql/
http://oai.rkbexplorer.com/sparql/
http://pisa.rkbexplorer.com/sparql/
http://ulm.rkbexplorer.com/sparql/
http://eurecom.rkbexplorer.com/sparql/
http://ibm.rkbexplorer.com/sparql/
http://resex.rkbexplorer.com/sparql/
http://ieee.rkbexplorer.com/sparql/
http://dotac.rkbexplorer.com/sparql/
http://laas.rkbexplorer.com/sparql/
http://eprints.rkbexplorer.com/sparql/
http://newcastle.rkbexplorer.com/sparql/
http://budapest.rkbexplorer.com/sparql/
http://deploy.rkbexplorer.com/sparql/
http://lod-cloud.net/southampton-ecs-eprints
http://roma.rkbexplorer.com/sparql/
http://curriculum.rkbexplorer.com/sparql/
http://irit.rkbexplorer.com/sparql/
http://nsf.rkbexplorer.com/sparql/
http://cordis.rkbexplorer.com/sparql/
http://courseware.rkbexplorer.com/sparql/
http://data.semanticweb.org/sparql
http://data.bibbase.org:2020/sparql
http://core.kmi.open.ac.uk/sparql
http://lov.okfn.org/endpoint/lov
http://lod-cloud.net/schemapedia
http://api.talis.com/stores/ordnance-survey/services/sparql

Fig. 5: Usage of datasets in the ‘Linking Open Data’ cloud by YASGUI users, anno 2014. Blue datasets appear in YASGUI queries, the size depends on the number of links stated in the Datahub.

Taking a closer look at the different endpoints, we divided them into a five categories (See Table 4b). To filter typographic errors, we reduce the list of 931 endpoints to a list of endpoints which only contain those on which more than 1 query was executed. This results in a list of 537 endpoints. For each of the endpoints in this filtered list, we check whether this endpoint is accessible. When the endpoint is accessible, we check whether it occurs in the CKAN catalog. The set of inaccessible endpoints are not part of the Linked Open Data cloud. However, this does not mean that they only contain private or closed data: users might store a copy of a CKAN dataset locally for analysis. Therefore, we analyze the namespaces in the corresponding queries of these endpoints: whenever a

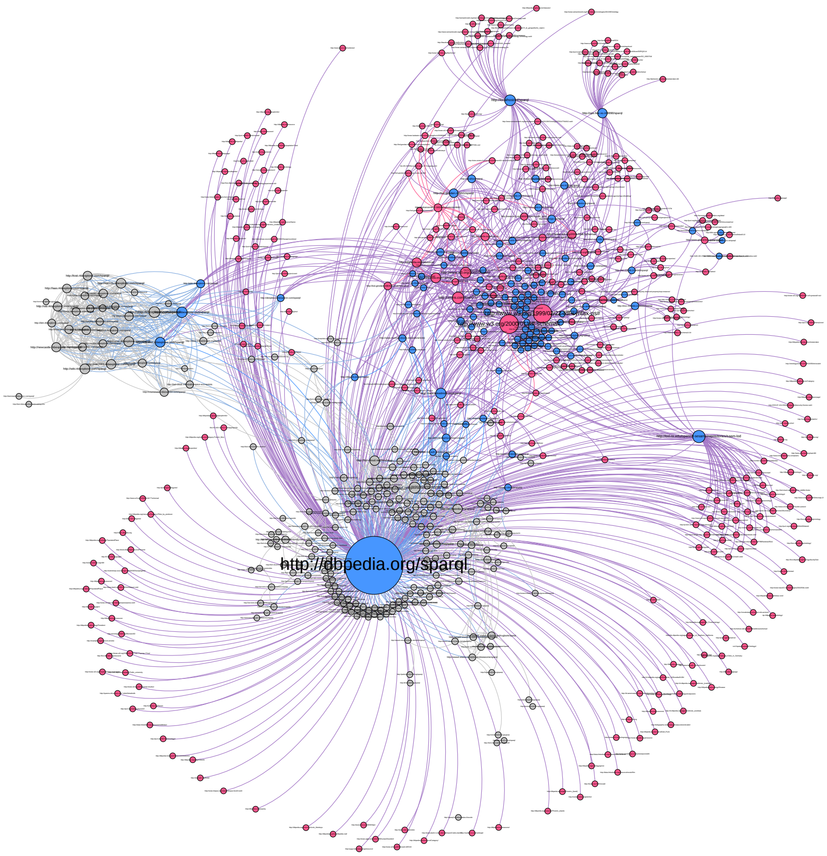


Fig. 6: Usage of datasets in the 'Linking Open Data' cloud by YASGUI users, and non-listed endpoints, related to namespaces, anno 2014. Red circles are namespaces, blue datasets appear in YASGUI queries, the size depends on the number of links stated in the Datahub, combined with the number of links with namespaces.

namespace does not occur in the prefix.cc[21] collection, we assume this endpoint contains private data. This gives us 105 endpoints, from which we can derive that 11.02% of all queries are executed on an endpoint containing private data. In other words, from the YASGUI usage perspective, 89% of the Linked Data Cloud is open, where the other 11% is closed.

Dataset Usage We can determine what part of each data set is touched by queries, by rewriting all SPARQL SELECT queries to CONSTRUCT queries. This gives us, for each pair of query and endpoint, the triples needed to answer

[21] See `http://prefix.cc` (6 May 2014)

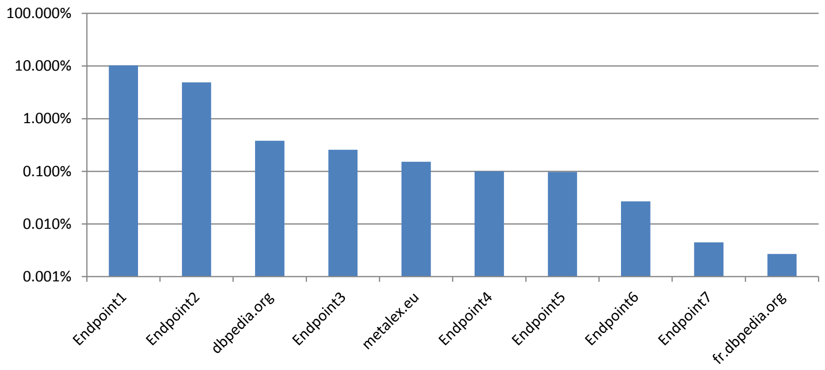


Fig. 7: Query coverage of top 10 used datasets in YASGUI (log scale). Endpoints not available through the Datahub are anonymized

the original SELECT query. We performed this analysis for the 10 most often used datasets that were accessible at the time of the experiment (see Figure 7). This shows that for most endpoints, less than 0.4% of the dataset is actually needed to answer our queries. DBpedia (the most popular endpoint) requires only 0.38% of its size, to answer 8179 queries.

Namespace Usage The query logs allow us to see what type of information from the Web of Data is used. Namespaces are good candidates to look at, as they reflect the use of often domain specific vocabularies. Table 2 shows the 10 most common namespaces used between all the queries. As expected, the RDF type and RDF schema namespaces are the most popular ones. Table 3 shows a similar list of prefixes, based on the page views of users on Prefix.cc until February 2011 [22]. Because no specific numbers are available (only a ranking of namespaces by page views), we compare the namespace rankings between Prefix.cc and YASGUI. Comparing both to the use of namespaces as tracked by LODStats (Table 4), we can see that DBpedia-based namespaces are more frequently used in queries than that they are reused across datasets. Additionally, the higher ranked RDF Schema namespace and presence of the OWL namespace in the YASGUI ranking, indicates that users rely more on schema information than data publishers.

Using the pairing of namespaces and datasets, we can create a map of the commonalities between datasets. Figure 6 shows a bipartite network of namespaces and datasets that groups together those namespaces and datasets that are most similar, purely based on network topology. Remember that the links in the LOD cloud diagram are based on the reuse of individual identifiers belonging to the datasets. Figure 6 links datasets that are similar at the schema level.

[22] See `http://richard.cyganiak.de/blog/2011/02/top-100-most-popular-rdf-namespace-prefixes/` (12 May 2014)

Namespace	%	#
http://www.w3.org/2000/01/rdf-schema#	16.5%	10.350
http://www.w3.org/1999/02/22-rdf-syntax-ns#	15.9%	9.962
http://dbpedia.org/property/	11.4%	7.142
http://dbpedia.org/resource/	11.0%	6.922
http://dbpedia.org/ontology/	10.9%	6.869
http://xmlns.com/foaf/0.1/	6.2%	3.882
http://dbpedia.org/	3.1%	1.968
http://www.w3.org/2001/XMLSchema#	2.6%	1.642
http://www.w3.org/2002/07/owl#	2.3%	1.437
http://www.w3.org/2004/02/skos/core#	1.6%	1.017

Table 2: YASGUI: Top 10 namespaces occurring in queries

Namespace	Prefix.cc	YASGUI
http://xmlns.com/foaf/0.1/	1	6
http://purl.org/dc/elements/1.1/	2	10
http://www.w3.org/1999/02/22-rdf-syntax-ns#	3	2
http://www.w3.org/2000/01/rdf-schema#	4	1
http://www.w3.org/2002/07/owl#	5	9
http://www.geonames.org/ontology#	6	13
http://www.w3.org/2003/01/geo/wgs84_pos#	7	14
http://www.w3.org/2004/02/skos/core#	8	10
http://dbpedia.org/property/	9	3
http://swrc.ontoware.org/ontology#	10	89

Table 3: Prefix.cc: Top 10 namespace rankings based on user page views

Namespace	LODstats	YASGUI
http://www.w3.org/1999/02/22-rdf-syntax-ns	23.7%	15.9%
http://www.w3.org/2000/01/rdf-schema	15.9%	16.5%
http://purl.org/dc/terms/	10.9%	1.5%
http://www.systemone.at/2006/03/wikipedia	6.0%	*none*
http://d-nb.info/standards/elementset/gnd	5.3%	*none*
http://www.w3.org/2004/02/skos/core	2.8%	1.6%
http://iflastandards.info/ns/isbd/elements	4.7%	0.00%
http://fao.270a.info/property/	2.1%	*none*
http://www.aktors.org/ontology/portal	2.1%	0.00%
http://schema.org	2.1%	0.3%

Table 4: LODstats: Top 10 namespaces based on occurrences in triples

Query Complexity Analysis Table 4a shows a number of statistics based on a total of 45.323 queries collected via YASGUI. After filtering invalid queries using the Jena[23] query parser, this number drops to 30.482 queries. This large number of invalid queries is partly due to the strict parsing of Jena. Some queries may not conform to the SPARQL standard, but return valid SPARQL results for certain endpoints regardless. For example, a query containing a 'bif:' URI, supported by Virtuoso endpoints, is marked as invalid. When we remove duplicate queries from the query set, 18.162 queries remain.

We observe that the majority of queries executed via YASGUI are `SELECT` queries. Both `ASK` and `DESCRIBE` queries amount to a fraction of the YASGUI query logs (1.59% and 0.74% respectively). We believe this shows that users prefer the more common `SELECT` keyword instead. Rather than the boolean value returned by an `ASK` query, the user may evaluate the query results from the `SELECT` query as-is. We expect this is due to the familiarity users have with SELECT queries; only a few of them will opt for an `ASK` or `DESCRIBE` query. Interestingly, the number of executed CONSTRUCT queries amounts to only 3.15%, which might indicate that data re-use via SPARQL queries is uncommon.

Another observation concerns the complexity of SPARQL queries. Table 4a shows that 54.35% of the queries contain one or more joins, and the most common triple patterns consists of VCC and VCV triple patterns. Such statistics can be used for optimizing man-made queries, and tell us more about how people query Linked Data.

When we take a closer look at the individual queries contained in the logs, we see that we can glean information about more than the queries only. First, we observe that 72.66% of executed queries are inefficient. As described in [17], the use of OPTIONALs in SPARQL is inefficient, and should be avoided whenever possible. If a triple pattern will always be bound for a given dataset, one should not use that triple pattern in an OPTIONAL. Using this rule of thumb, we analyzed all the queries which use an OPTIONAL and which relate to a public endpoint. For the result set of these queries, we analyze whether the number of returned bindings for the OPTIONAL triple patterns differ between query solutions. When the number does not differ, then the OPTIONAL was not used appropriately, and the query would have been more efficient with either the OPTIONAL triple pattern removed, or added to the query as regular triple pattern.

5 Conclusion

This paper uses YASGUI as a means to analyze the Web of Data. Given the richness of features compared to other SPARQL clients, YASGUI is rapidly becoming a popular interface to the Web of Data, positioning itself as a dataset independent data collection point which can act as a kind of observational lens. We are aware the results presented in this paper are not fully representative and unbiased. However, we argue exactly the same for alternative dataset statistics: these are either based on dataset catalogs, or on an opt-in basis, making these statistics incomplete.

[23] See `http://jena.apache.org/` (6 May 2014).

Only 1 year after the release of YASGUI, we are already able to analyze a large amount of statistics on the Web of Data. This gives unprecedented insight into how we actually use the Linked Data cloud, and what part of the Linked Data cloud we use. Using the collected data, we were able to analyze the efficiency of queries, what part of the used Linked Data cloud is open or closed, what part of these datasets we use, the complexity of queries, and the shared use of namespaces over all the endpoints. With an increase in uptake of YASGUI, we will be able to make these claims even stronger, and we will be able to understand the use of Linked Data even better. More data allows us to recognize more fine-grained patterns, e.g. to identify a relation between the structure of a dataset and its queries, which categories of queries exist, and how these query categories relate to typical tasks.

To conclude, this paper introduces a tool, dataset and methodology that increase our knowledge of the use of Linked Data. It allows for analyzing the Linked Data cloud in the broadest sense: what datasets exist, how are they used, and for what purpose? The amount of data we gathered in this short period of time, and the increasing uptake of YASGUI, promises an even clearer picture of Linked Data in the future.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: The semantic web, pp. 722–735. Springer (2007)
2. Auer, S., Lehmann, J., Hellmann, S.: Linkedgeodata: Adding a spatial dimension to the web of data. In: The Semantic Web-ISWC 2009, pp. 731–746. Springer (2009)
3. Berendt, B., Hollink, L., Luczak-Rösch, M., Möller, K., Vallet, D.: Proceedings of USEWOD2013 - 3rd international workshop on usage analysis and the web of data. In: 10th ESWC - Semantics and Big Data, Montpellier, France (2013)
4. Bizer, C., Jentzsch, A., Cyganiak, R.: State of the lod cloud. Version 0.3 (September 2011) 1803 (2011)
5. Bizer, C., Seaborne, A.: D2rq - treating non-rdf databases as virtual rdf graphs. World Wide Web Internet And Web Information Systems p. 26 (2004)
6. Borsje, J., Embregts, H.: Graphical query composition and natural language processing in an rdf visualization interface. In: Erasmus School of Economics and Business Economics, Vol. Bachelor. Erasmus University, Rotterdam (2006), http://themis.jesdesign.nl/~jaborsje/personal/publications/bachelor-thesis.pdf
7. Broekstra, J., Kampman, A., Van Harmelen, F.: Sesame: An architecture for storing and querying RDF data and schema information (2001)
8. Buil-Aranda, C., Hogan, A., Umbrich, J., Vandenbussche, P.Y.: Sparql web-querying infrastructure: Ready for action? In: The Semantic Web–ISWC 2013, pp. 277–293. Springer (2013)
9. Callahan, A., Cruz-Toledo, J., Ansell, P., Dumontier, M.: Bio2rdf release 2: Improved coverage, interoperability and provenance of life science linked data. In: The Semantic Web: Semantics and Big Data, pp. 200–212. Springer (2013)
10. Campinas, S., Perry, T.E., Ceccarelli, D., Delbru, R., Tummarello, G.: Introducing rdf graph summary with application to assisted sparql formulation. In: Database

and Expert Systems Applications (DEXA), 2012 23rd International Workshop on. pp. 261–266. IEEE (2012)

11. Demter, J., Auer, S., Martin, M., Lehmann, J.: Lodstats – an extensible framework for high-performance dataset analytics. In: Proceedings of the EKAW 2012. Lecture Notes in Computer Science (LNCS) 7603, Springer (2012), 29acceptance rate
12. Gallego, M.A., Fernández, J.D., Martínez-Prieto, M.A., de la Fuente, P.: An empirical study of real-world sparql queries. In: 1st International Workshop on Usage Analysis and the Web of Data (USEWOD2011) at the 20th International World Wide Web Conference (WWW 2011), Hydebarabad, India (2011)
13. Harris, S., Lamb, N., Shadbolt, N.: 4store: The design and implementation of a clustered RDF store. In: 5th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2009). pp. 94–109 (2009)
14. Hoekstra, R., Groth, P.: Linkitup: Link discovery for research data. In: Discovery Informatics: AI Takes a Science-Centered View on Big Data, AAAI Fall Symposium Series (2013)
15. Hogan, A., Umbrich, J., Harth, A., Cyganiak, R., Polleres, A., Decker, S.: An empirical survey of linked data conformance. J. Web Sem. 14, 14–44 (2012)
16. Hogenboom, F., Milea, V., Frasincar, F., Kaymak, U.: RDF-GL: a SPARQL-Based graphical query language for RDF. In: Emergent Web Intelligence: Advanced Information Retrieval, pp. 87–116 (2010), `http://link.springer.com/chapter/10.1007/978-1-84996-074-8\_4`
17. Loizou, A., Groth, P.T.: On the formulation of performant sparql queries. CoRR abs/1304.0567 (2013)
18. Möller, K., Hausenblas, M., Cyganiak, R., Handschuh, S.: Learning from linked open data usage: Patterns & metrics. In: WebSci10: Extending the Frontiers of Society On-Line. pp. 1–9 (2010)
19. Möller, K., Heath, T., Handschuh, S., Domingue, J.: Recipes for semantic web dog food. the eswc and iswc metadata projects. In: The Semantic Web, pp. 802–815. Springer (2007)
20. Noy, N.F., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D.L., Storey, M.A., Chute, C.G., et al.: Bioportal: ontologies and integrated data resources at the click of a mouse. Nucleic acids research 37(suppl 2), W170–W173 (2009)
21. Openlink Virtuoso: Universal server platform for the real-time enterprise (2009), `http://www.openlinksw.com/`
22. Picalausa, F., Vansummeren, S.: What are real sparql queries like? In: Proceedings of the International Workshop on Semantic Web Information Management. p. 7. ACM (2011)
23. Rietveld, L., Hoekstra, R.: Yasgui: Not just another sparql gui. In: Proceedings of the Workshop on Services and Applications over Linked APIs and Data (SALAD2013) (2013)
24. Rietveld, L., Hoekstra, R.: Man vs. Machine: Differences in SPARQL Queries. In: ESWC, 4th USEWOD Workshop on Usage Analysis and the Web of Data (2014)
25. Smart, P.R., Russell, A., Braines, D., Kalfoglou, Y., Bao, J., Shadbolt, N.R.: A Visual Approach to Semantic Query Design Using a Web-Based Graphical Query Designer. In: Knowledge Engineering: Practice and Patterns, pp. 275–291 (2008)
26. Tummarello, G., Delbru, R., Oren, E.: Sindice. com: Weaving the open linked data. In: The Semantic Web, pp. 552–565. Springer (2007)
27. Zviedris, M., Barzdins, G.: ViziQuer: a tool to explore and query SPARQL endpoints. In: The Semanic Web: Research and Applications, pp. 441–445 (2011), `http://link.springer.com/chapter/10.1007/978-3-642-21064-8\_31`